C and Lisp Languages

In today's world everybody at least once in their lives heard the word computer. However, how many percentages of people knows its definition and even if we go details who knows meaning of terms computing, software, hardware, programming, and programming languages? These terms are main parts of a computer. This essay is going to give brief information about what is a programming language then, it will go into details of programming languages C and LISP and show their similarities, differences, advantages, and disadvantages among them.

First of all, what is a programming language? There are a lot of different answers for this question. Those answers can be summed up as, it is a conceptual universe which helps people to communicate with and give instructions to a computer so that it can perform given tasks. With those tasks people can handle their jobs and solve their problems in a fast and efficient way. Of course, a programming language must have some set of rules about syntax and semantics. Because even in daily life conversations people might understand different meanings from the same sentence or quote and to computer be able to perform given tasks in a correct way, instructions must be precise. Nowadays, there are thousands of programming languages which satisfy these conditions. Those languages are divided into 2 levels according to their communications between hardware. These levels are low-level and high-level. Low-level has 2 examples, machine code and assembly language. In machine code which is the lowest level language we have only ones (1) and zeros (0) to be able to communicate with machines. Assembly language works like a bridge between machine code and high-level languages. High-level languages are more human readable languages and machine independent. There are programming paradigms to classify each language according to their features. A paradigm is a model for helping us to understand concepts accepted by sector. Most languages are not pure according to their paradigms. They can contain different paradigms features but some paradigm features are distinguished. In the following part this essay will talk about procedural programming language C and functional programming language LISP.

C is a general-purpose programming language which is developed at Bell Laboratories by Dennis Ritchie in 1972. They needed programs that will help UNIX systems but in those days

programming languages were not much capable and efficient for writing operating systems. Therefore, he decided to develop a new programming language which is called C. C is a successor of B programming language. Day by day C got popular and even it is over 55 years old it is still one of the top programming languages of programming industry. It is in 4th place in IEEE's "Top Programming Languages 2023" spectrum. 4th place is not bad. When you look at the top three which are Python, Java, and C++, they are influenced by C language while they were being developed. C's simple design, speed, efficiency, performance, and close relationship with computers' hardware makes it an ideal choice against other languages. By using pointers, C even give programmers low-level access to memory. These are the highlights of popularity of C programming language. C is a procedural language which specifies a series of steps for a program to solve the problem. Even though it does not belong to functional programming paradigm it has all executable codes in subroutines which also known as functions. In 1978 Brian Kernighan and Dennis Ritchie together wrote a book called "The C Programming Language". For many years people come and check this book when they need something related to C language. To make sure there was a standard, machine independent definition of the language, C has been standardized by American National Standard Institute (ANSI) and the International Organization for Standardization (ISO) since 1989. Thanks to C's compatibility we can write too many different types of programs on different platforms. We learnt little things about C language now let's get some brief history and features of LISP language.

Lisp is developed by John McCarthy while he was at Massachusetts Institute of Technology (MIT) in 1958. Its first implementation is done by Steve Russel on an IBM 704 computer using punched cards and they find out this was a Lisp interpreter.. Then they realized that Lisp's eval functions can be implemented to machine code. As the time passes Lisp is improved and extended it becomes a big family at the end. It has a lot of dialects. The best-known general-purpose dialects are Common Lisp, Scheme, Racket, and Clojure. While Lisp was extending itself it improved not only itself but also helped to computer science's improvement. It pioneered many ideas in computer science. For example, tree data structures, automatic storage management, dynamic typing, higher order functions and the read-eval-print loop, etc. Lisp is an acronym for LISt Processing and its one of

the major data structures are linked lists. Lisp has a distinctive, fully parenthesized prefix notation. With this feature there will be no ambiguity about order of operations or operands. Lisp was originally developed as a practical mathematical notation for computer programs and symbolic computation and manipulation. It was really helpful for artificial intelligence researchers. So far, this essay made some brief introduction about what it is going to tell next. Now, the question is what are the differences between C and Lisp?

Firstly, there is an obvious and expected syntax and semantic difference between C and Lisp. To be able to see those differences clearly we are going to itemize them.

- In C each line of instruction has to be finished with a semicolon(;). Lisp does not need something like that.
- While in most of the programming languages functions are called by their names and following parentheses. In Lisp it is not the same. For example, to be able to write “Hello, World!” you need to write following code (print “Hello, World!”). We can define strings in Lisp in between two quotation marks however in C we need to use character arrays.
- Lisp has dynamic typing. In contrast, C has static casting. In dynamic typing type checking is done at runtime. If the programmer does not handle this clearly this might end up with a runtime error. In static typing type checking is done at compile time. Programmers have to specify the type of a variable.
- In Lisp there are two data types. Scalar types are used to store single values. Examples are number types, character, etc. Second one is data structures, and they are used to store multiple values. We can give arrays and vectors as examples to it. In C there are four types of data types. First one is primitive data types. It is the basic data type in C. It includes integers, floats, characters, etc. C language supports both signed and unsigned literals. This gives programmers more flexibility and bitwise operations. Second is derived data types. This type gives programmers the ability to handle heterogenous data, directly modify memory and build complicated data structures. Examples of this data type are array, functions, pointers, etc. Third data type is user defined data types. These are the ones that can be generated by users by

using C's abilities. Structure, enumeration, typedef, and unions can be given as examples. The last and the fourth one is void data types. Void type is used when there is lack of a particular type. Void type is generally used as, function's return types, pointers, function parameters.

- Another difference is commenting. To make the written codes more readable and understandable programmer of the code needs to use comments. When other programmers see the code and if they can not understand, they will check comments for clarification. Therefore, comments are very important things for coding. In C, for comments programmers use // for single line comments, and use /**/ for multiple line comments. In Lisp semicolons(;) are used for comment statements. Using one semicolon is enough for normal case but as a programmer you can use semicolons as much as you want.
- In computers every file has an extension at the end of their file names. For C files this can be .h for header files, and .c for source code files. For Lisp files extension will be .lisp.
- In C, files can be compiled separately or linked together. To be able to do linked compile, first files should be compiled and as its result we will get object files. Then we will link those object files into an executable file. When we run this file we will see output of our codes. In Lisp, there is no linking system but there are functions for it as part of REPL(read - eval - print loop). LOAD function is one of those.

Secondly, Lisp has garbage collection system while C does not. High-level languages can be divided into two parts as compiled and interpreted. Compiled ones are machine friendly they are directly converted to machine code then they run. Interpreted programs are executed by a runtime. This runtime makes automatic memory management. There is a memory space which is allocated before by runtime itself. While runtime meets memory requests that space is being used. That space is called managed heap. Garbage collection (GC) is a part of runtime. And GC system works for both memory allocation and release of that memory. It helps programmers a lot because without an automatic garbage collector, programmer would have to deal with all those allocations, release and they have to think about memory in terms of sufficiency. Therefore, having automatic garbage collection system is an advantage. Although there is no such thing in C, it gives low-level memory

access opportunity to programmers. It has 4 library functions to handle dynamic memory allocation. To be able to use those functions you need to add #include <stdlib.h> at the beginning of your code. First of these functions is malloc() it is an acronym for memory allocation. It allocates specified size memory space for the process and returns a void pointer. This pointer can be converted to any type of pointer. For an integer type memory allocation's syntax is as follows, ptrX = (int*) malloc(100 *sizeof(int));. In this way it allocates 400 bytes of memory because an integer keeps 4 bytes of memory space in the memory. To be able to reach that space we can use ptrX, it has that memory space's first byte's address in it. If there is not enough space for allocation, function returns NULL. Second function is calloc() which is an acronym for contiguous allocation. As it is seen in the name, it allocates memories as contiguously. For example, for ptrX = (int*) calloc(100, sizeof(int)); it will allocate contiguous space for 100 elements of type int. calloc() differs from malloc() in two ways. One is initialization, calloc() puts zero inside all allocated memory spaces. Two is syntax, calloc() has two arguments meanwhile malloc() has one. Also, malloc() is faster than calloc(). Third function is called as free() for deallocation purposes. Just because malloc() and calloc() does not automatically release memory space that they have allocated C has free() function for it. To free an allocated space, you need to just pass that address. For above cases this is done as free(ptrX);. The last and the fourth function is realloc() which is an acronym for re-allocation. Let's say you have allocated memory via malloc() or calloc() but later on the amount of memory you allocated is not sufficient. Therefore, you decided to allocate more memory. In this case realloc() will be your choice to dynamically re-allocate memory. Let's say you allocated memory for 100 integers with malloc() like we have seen above. Now, you need to allocate for 120 integers in total. The syntax for it will be ptr = realloc(ptr, 120*sizeof(int));. Now, you increased your allocated memory from 400 bytes to 480 bytes without touching your previous data. If there is no enough space for realloc() it will return NULL. As you can see, with an automatic garbage collector these jobs are done so easily.

If we check these languages according to their fields of usage there is another diversity between them. Lisp is more suitable for artificial intelligence and quantum computing. On the other hand, C is used for embedded systems, operating systems, etc. For example, one of the Lisp family

member Common Lisp is being used in Grammarly which is a grammar checking startup. Its grammar engine is written in Common Lisp. It finds instances of incorrect tenses and suggests more precise synonyms for common words. Another Lisp example is SISCOG. SISCOG's Common Lisp rail scheduling system moves millions of passengers across Europe every day. For C's embedded system usage if we try to write all of them, there will be thousands of pages. Therefore, we can sum up it in a general form as C is used in daily life IoT devices, facilitator technologies for traffic management, etc. There are also operating systems that while they were written C is used. The UNIX, Windows and MacOS's kernels are fully written in C. Also, operating systems that are used in mobile phones, smartwatches, etc. are also written in C. Supercomputers and microcontrollers are other examples of C's usage field.

Up to now this essay talked about the differences between C and Lisp. Now, it is time to see similarities.

- C and Lisp both supports recursion. Recursion can be briefly defined as a function's ability to call itself. Recursion adds clarity and sometimes reduces the time needed to write and debug the code. Also, performs better in solving problems based on tree structures. In contrasts, it has disadvantages too. Recursion usually slower because it deals with stack and uses more memory for the stack.
- C and Lisp have common operators which are arithmetic, bitwise, comparison, and logical operators. Arithmetic operations are basic mathematical operations. Bitwise operations work on bits and perform bit-by-bit operations. Comparison operators are, also known as relational operators, the ones that takes two numbers and compare them. Logical operators are and, or, and not operator. Syntax is different in C but logic of use behind is the same.
- Both languages have preprocessor. In computer science a preprocessor is a program that processes its input data to produce output that is used as input in another program. C preprocessor takes lines beginning with # as directives. This is also called as

macros. Unlike C preprocessor macros, in Lisp the macros are Lisp functions and so can exploit the full power of Lisp.

- In both languages we can pass arguments as pass by value. However, C has additional properties. In C arrays are passed as pointers. A Pointer has address of first item. Pass by reference is simulated in C by explicitly.
- Both languages have rich libraries for additional functionality. C also has external libraries too.

To sum up, C and Lisp are two of the oldest programming languages. Their contribution to the computer science world is priceless. Without these two languages would the computer industry can improve this much we can not know. As we mention at the beginning C is a structural language that belongs to procedural paradigm. It is very useful for low-level programming, designing systems, and communication with hardware. Although, C helped for improvement of computer science by itself, it also contributed other programming languages development and those languages are also dominating the industry. On the other hand, Lisp is another oldest language which belongs to functional paradigm. It is very convenient for artificial intelligence, symbolic design, and quantum computers. As time passed, it became a big language family. And still serves the programming industry since 1958. As you can see, both languages have advantages and disadvantages according to usage field. It is up to programmers to use which one. With these languages, we can say that improvement in programming world will not stop, and it is inevitable.

RESOURCES

- https://en.wikipedia.org/wiki/Lisp_(programming_language)
- https://en.wikipedia.org/wiki/C_(programming_language)
- https://stackoverflow.com/questions/1517582/what-is-the-difference-between-statically-typed-and-dynamically-typed-languages

- https://learn.microsoft.com/en-us/dotnet/standard/garbage-collection/fundamentals
- https://www.geeksforgeeks.org/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/
- https://www.javatpoint.com/data-types-in-c
- https://lisp-lang.org/
- https://kuleuven-diepenbeek.github.io/osc-course/ch2-c/intro/#:~:text=The%20UNIX%2C%20Windows%20and%20OSX,build%20on%20top%20of%20C.
- https://www.tutorialspoint.com/cprogramming/c_operators.htm
- https://www.tutorialspoint.com/lisp/lisp_operators.htm
- https://www.cl.cam.ac.uk/teaching/1213/ConceptsPL/l4.pdf